MM1 模擬:

一、模擬內容:

模擬 M/M/1 的排隊過程，我們把一個客人到店視為一個事件:

- 該系統一次只能處理一個客人
- 每位客人到店的時間間隔呈現 exponential distribution，參數為 lambda，白話文是單位時間來了 Lambda 位客人，也代表客人與下位客人到訪時間差可以用(-1/lambda) * log(Ui) 來表示。另外，server 服務每一位客人所需的時間也呈現 exponential distribution，參數為 miu，表示單位時間平均可以服務 miu 位客人，也代表服務每位客人的時間可以用(-1/miu) * log(Ui) 來表示。

- 客人到店後，有兩種情況，一種前方沒有客人正在被服務，那他就不用等，waiting time = 0，如果前方有客人，就要等到該人離開後，才能開始被服務，因此會需要一段等待時間: waiting time =「前位客人的離開時間 - 自己到達的時間」。這個系統會有一個截止時間 T，也就是 T 時間之後就不允許有額外進店的客人，但是會等到最後一個客人被服務完，系統才會關閉。

我們想去探討:

在不同 lambda，miu 的情況下，客人所需的期望等待時間(Expected waiting time 是多久)。

以及在這過程中，店員有多少比例的時間是需要服務客人的(Utilization: 系統真正運作時間/ 系統開機時間)。

Sample Mean 的概念:

然而，由於 MM1 這個系統在不同次模擬中(就算同樣 lambda，miu)，各個客人所需等待的時間的分布都不盡相同，也無窮無盡。我們其實不可能知道實際 MM1 在一個 Lambda,miu 下的 population mean waiting time。因此，我們只能透過模擬多次，將每一次的模擬得到的「平均等待時間」作為一個 Xi，透過多次模擬，取得多個樣本 X1~ Xn，再抓 X1 ~ Xn 作樣本，利用樣本平均，去估計如果一個 MM1 系統在特定 lambda，miu 下，真正(當無窮無盡的客人的平均)等待時間(population mean waiting time)，也因此一個樣本的 sample size 會是 n。

信賴區間:

並且，我們也不是真正確定自己的估計值就完全是真正的平均等待時間，因此加入信賴區間的概念，樣本數 n 為模擬次數，

而 sample variance s^2 為 $\frac{\sum_{i=1}^{n}(X_i - X_{mean})^2}{n-1}$ 。

95% Confident Interval 為: $X_{mean} \pm 1.96\frac{s}{\sqrt{n}}$ 。

二、Code:

MM1 一個 simulation 模擬的方法是:

先產生一個 arrival list: arr[]，記錄所有客人到店的時間(第一位客人的到店時間為 0)

```
#GENERATE ARRIVAL LIST
while(t <= T):
    arr.append(t)
    t += -(1/Lmd)*log(rnd.uniform(0,1))
```

產生一個 Departure list: dep[]，記錄所有客人離店時間

在產生第一個人的離店時間(即他的 service time)後，站在現在客人(eg. 第一位)的角度，去看下一位客人(eg. 第二個人)的到店時間 arr[j+1]是否比現在客人的離店時間 dep[j]早?

```python
#GENERATE DEPARTURE LIST
    #first person
#first person
st  = -(1/Miu)*log(rnd.uniform(0,1))
st_list.append(st)
dep.append(st)
j = 0


while( j + 1 < len(arr)): #j is index
    st = -(1/Miu)*log(rnd.uniform(0,1))
    st_list.append(st)
    if(dep[j] >= arr[j + 1]):
        dep.append(dep[j]  + st)
    else:
        dep.append(arr[j + 1]  + st)
    j += 1
```

如果比較早代表下一位先到，到時前方還有客人，所以她的離店時間 dep[j+1]就是上一位客人離店時間 dep[j] + service time(st)。

```python
  if(dep[j] >= arr[j + 1]):
      dep.append(dep[j]  + st)
```

如果比較晚，代表下一位客人到的時候，前方已經沒有客人，所以離店時間就是到店時間加上服務時間 arr[j+1] + st。

```python
dep.append(arr[j + 1]  + st)
```

St_list[] 會去記錄每次的 service time st。

有了 dep[], arr[] 計算 waiting time:

比較現有客人的到店時間比上一位客人離店時間早，代表要等到上一位客人離店後才被服務所以 waiting time = dep[i - 1] - arr[i]。反之，則不用等，所以 waiting time = 0。

```python
#CALCULATE WAITING TIME OF EACH PERSON BASED ON ARR AND DEP LIST
waiting_list = []
waiting_list.append(0) #first person
for i in range(1,len(arr)):
    wt = 0 if arr[i] >= dep[ i - 1] else dep[i - 1] - arr[i]
    waiting_list.append(wt)
```

增加信賴區間:

由於固定系統接受客人的時間 T =500，我們可以期待總人數 = lambda * T，如果太大，模擬的時間會過長，所以這邊觀察 lambda = 0.1 ~ 1 每個間隔 0.01 的情況下，在 miu = 0.5 / 0.75 / 1 時的等待時間狀況。

NUM_SIM: 模擬次數

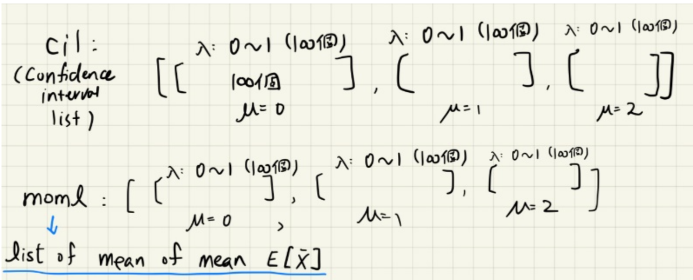
Mean = Xi

Mean_wts: sample mean $\overline{X}$

Moml: 有三個 sublist，每個 sublist 對到一 miu 值底下不同 lambda 的 E[$\overline{X}$]

Cil: 有三個 sublist，每個 sublist 對到一 miu 值底下不同 lambda 的

$1.96 * \frac{sample\ standard\ deviation}{NUM\_Sim - 1}$

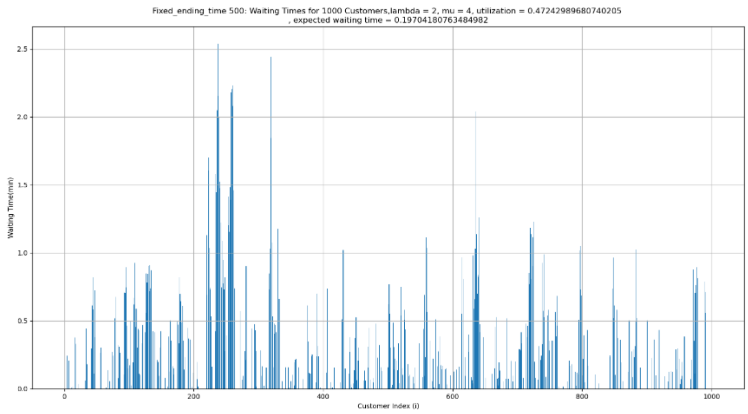



```python
NUM_SIM = 250
miu_list = [0.5,0.75,1]
lamda_list = np.arange(0.1, 1.01, 0.01)
moml = [[],[],[]]
cil = [[],[],[]]
for m, miu in enumerate(miu_list):
    for l, lamda in enumerate( lamda_list ):
        mean_wts = []
        for s in range(NUM_SIM):
            mean = mm1_cal_wt( lamda, miu ) #mm1_cal_wt:a function that simulates a single round of mm1 and returns the average waiting time (mean)
            mean_wts.append(mean) # mean_wts: a list of mean Xi
        mean_wts = np.array(mean_wts)
        moml[m].append(np.mean(mean_wts)) # moml (mean of mean list): a list of mean of mean: add one element for one lamda, miu pair that's gone through NUM_SIM of simulation
        cil[m].append( 1.96 * sqrt(np.var(mean_wts, ddof=1))/sqrt(NUM_SIM)) #np.var(mean_wts, ddof=1) is the sample variance, cil is the  list of (1.96 * sample_std/ n-1)
```

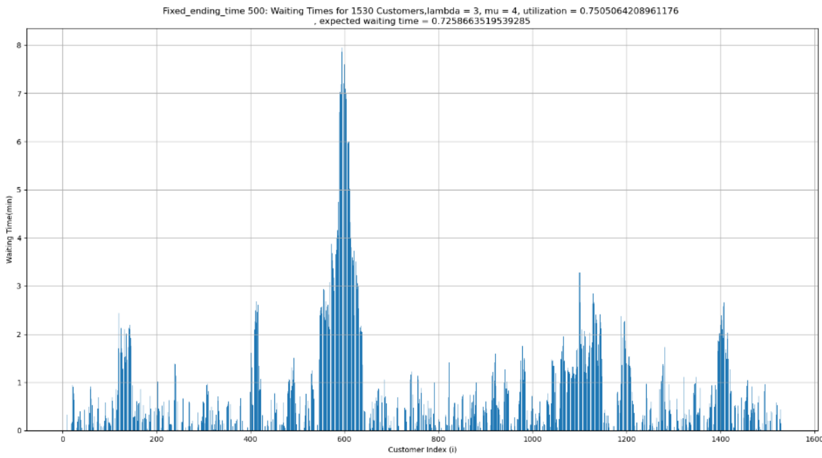
作圖：

```
moml = np.array(moml)
cil = np.array(cil)
# Plot the lines on the same plot
plt.plot(lamda_list, moml[0] , label='miu = 75')
plt.fill_between(lamda_list, moml[0] - cil[0], moml[0] + cil[0], color='gray', alpha=0.3)
plt.plot(lamda_list, moml[1] , label='miu = 100')
plt.fill_between(lamda_list, moml[1] - cil[1], moml[1] + cil[1], color='purple', alpha=0.3)
plt.plot(lamda_list, moml[2] , label='miu = 125')
plt.fill_between(lamda_list, moml[2] - cil[2], moml[2] + cil[2], color='green', alpha=0.3)

# Add labels and a legend
plt.xlabel('lambda')
plt.ylabel('waiting time')
plt.title(f'{NUM_SIM} simulation')


# Show the plot
plt.show()
```

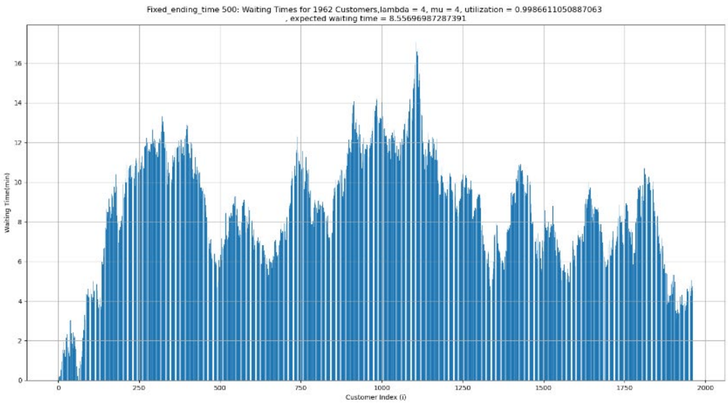
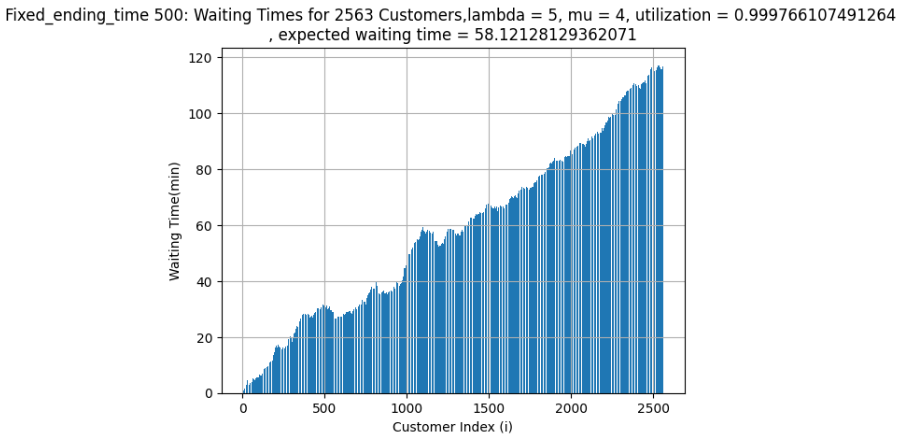
三、模擬結果：

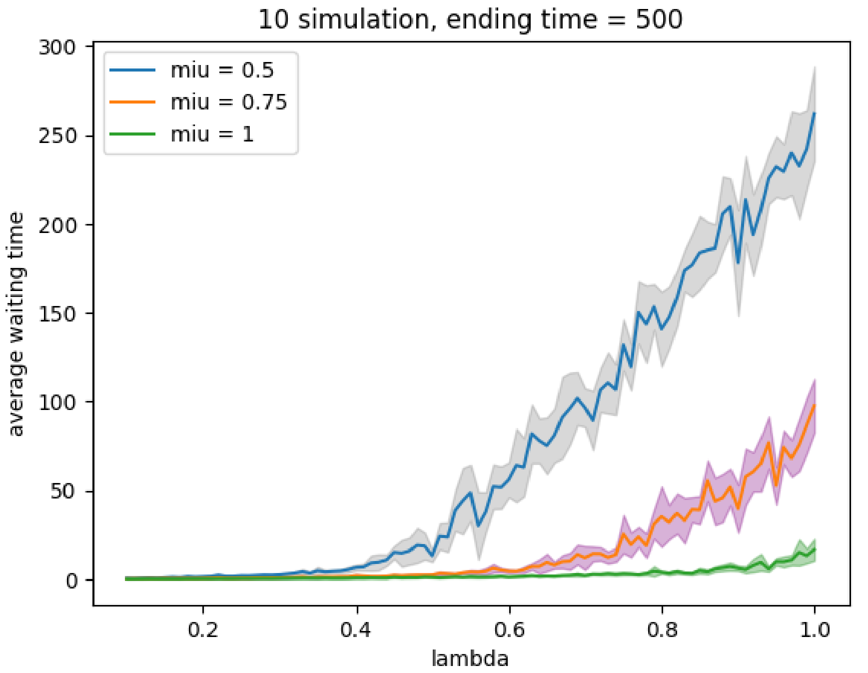


等待時間分佈：Lambda: 2, Miu: 4， Expected waiting time: 0.197，Utilization: 0.472

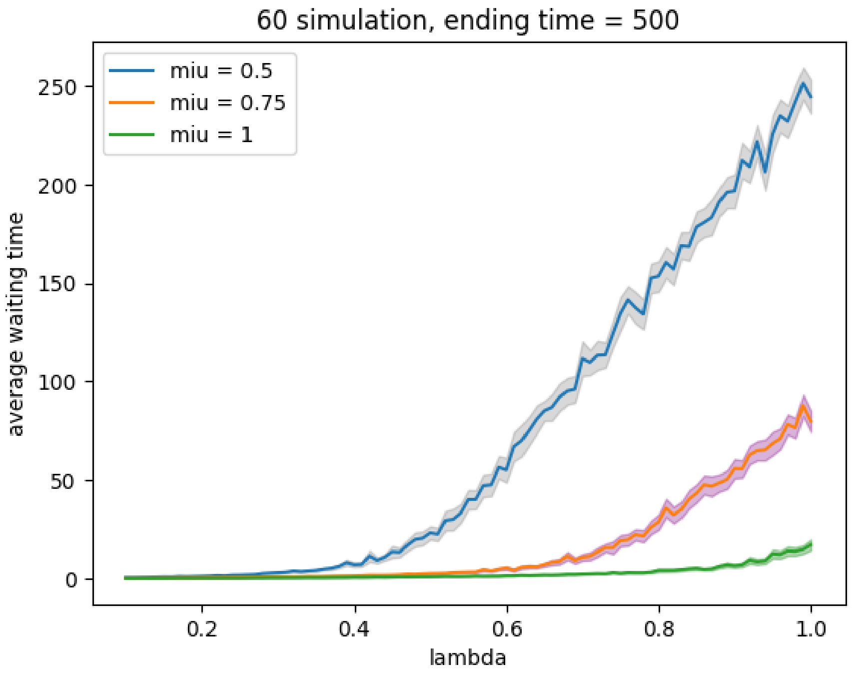


等待時間分佈：Lambda: 3, Miu: 4， Expected waiting time: 0.7258，

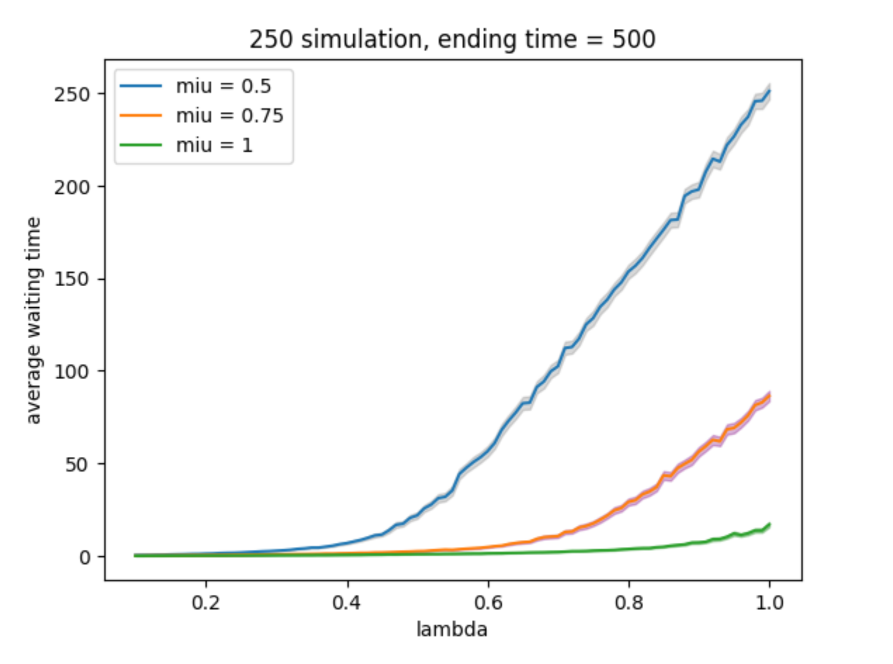
Utilization: 0.7505

等待時間分佈: Lambda: 4, Miu: 4， Expected waiting time: 8.557， Utilization: 0.998

等待時間分佈: Lambda: 5, Miu: 4， Expected waiting time: 58.12， Utilization: 0.9997


10 simulation, ending time = 500
miu = 0.5
miu = 0.75
miu = 1
average waiting time
lambda
0
50
100
150
200
250
300
0.2
0.4
0.6
0.8
1.0

10 次模擬


60 simulation, ending time = 500
miu = 0.5
miu = 0.75
miu = 1
average waiting time
lambda
0
50
100
150
200
250
0.2
0.4
0.6
0.8
1.0

60 次模擬



250 次模擬

觀察模擬結果可以發現當 miu 值固定時，隨著 lambda 上升，等待時間，與 utilization 均會上升。仔細看可以發現 utilization 基本上接近 lambda/ miu。

當 lambda = miu 時，等待時間分布突變的密集，表示此時開始大家都需要等，平均等待時間上升，到 lambda>miu 時，越後面的等待時間呈現上升趨勢，沒有下降趨勢，表示如果越往後的客人可能會需要等待無限長的時間，系統不穩定。當 lambda > miu 時，表示單位時間能夠服務的客人少於來的客人數 lambda，無法及時消化客人使得會不斷累積客人，queue 越來越大，導致最後系統不穩定。

觀察 utilization: 在 lambda < miu 時，約等於 lambda/ miu，當 lambda == miu 時，utilization 接近 1，當 Lambda > miu 時，utilizaiton 更接近 1 或有些模擬中跑出 1 的結果，這時代表系統隨時都在服務客人，表示有些客人要等非長久，這種現象並不理想，系統通常會在 efficiency 和 utilization 中取的平衡。

呈現 waiting time 的方式，橫軸放的是 customer index，依照先後由左至右排列，這樣可以看到不同時間點進店的客人的等待時間的變化趨勢，如果我們用 histogram 觀察等待時間的次數分佈，便看不出客人進店的先後順序，使得無法觀察前面進店與後面進店客人的區別。

觀察有信賴區間的圖，可以發現平均等待時間從水平到開始上揚的點，接近 lambda = miu 的附近。當模擬次數增加時，可以發現信賴區間的]範圍變小

了，原因為根據信賴區間的公式$1.96 * \frac{sample\ standard\ deviation}{NUM_Sim - 1}$，此時 Num_Sim 變大導致區間變小，表示估計值更準。

最後，觀察藍綠紅三條線之間的區別可以發現，綠線的信心區間比其他小，原因猜測可能是因為 LAMBDA 在最大時 1，才約等於 MIU，表示從頭到尾系統都沒有爆掉，使等待時間的變化程度不會這麼高，因此估計出的信賴區間也不會那麼大。